EUROPEAN PATENT APPLICATION

(72) Inventor: Naffziger, Samuel D.
Fort Collins, CO 80525 (US)

(74) Representative:
Schoppe, Fritz, Dipl.-Ing.
Schoppe & Zimmermann
Patentanwälte
Postfach 71 08 67
81458 München (DE)

(54)   **Cache memory with reduced access time**

(57)   A cache with a translation lookaside buffer
(TLB) (210) that eliminates the need for retrieval of a
physical address tag from the TLB when accessing the
cache. The TLB includes two content addressable
memories (CAM's) (206, 208). For each new cache line,
in the tag portion of the cache (204), instead of storing
physical tags, the cache stores vectors called physical
hit vectors. Physical hit vectors are generated by a first
TLB CAM (206). Each physical hit vector indicates all
locations in the first TLB CAM containing the physical
tag (203) of the cache line. For a cache access, a sec-
ond TLB CAM (208) receives a virtual tag (202) and
generates a vector called a virtual hit vector (214). The
virtual hit vector indicates the location in the second
TLB CAM of the corresponding virtual tag. Then,
instead of retrieving and comparing physical tags, the
cache compares a virtual hit vector to a set of physical
hit vectors without having to retrieve a physical tag. As a
result, one operation is eliminated from a time critical
path, reducing the access time. For caches having vari-
able page sizes, an additional CAM structure (220)
stores page offset bits (224) and corresponding bit
masks from the operating system. Page offset bits are
then used to further qualify comparison of virtual hit
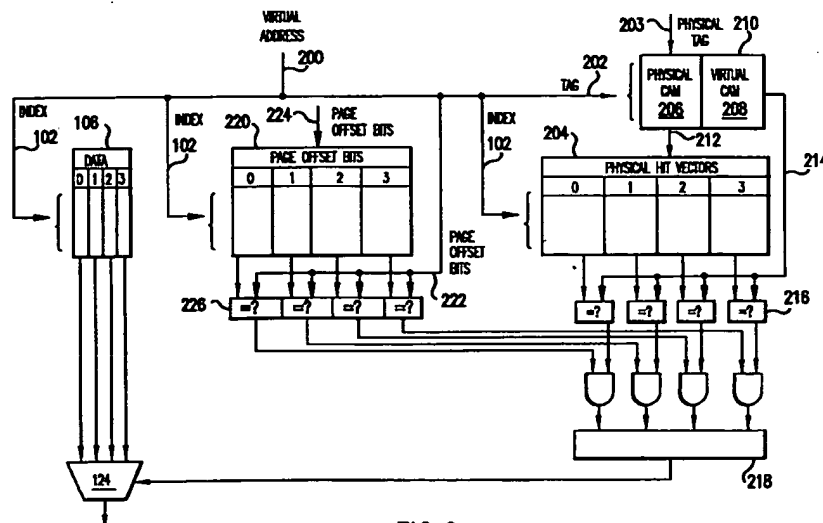vectors and physical hit vectors.

FIG.2

EP 0 911 737 A1

Description

FIELD OF INVENTION

[0001] This invention relates generally to computer memory systems and more specifically to improving access time in cache memory.

BACKGROUND OF THE INVENTION

[0002] Most computer systems employ a multilevel hierarchy of memory systems, with relatively fast, expensive, limited-capacity memory at the highest level of the hierarchy and proceeding to relatively slower, lower cost, higher-capacity memory at the lowest level of the hierarchy. Typically, the hierarchy includes a small fast memory called a cache, either physically integrated within a processor integrated circuit or mounted physically close to the processor for speed. There may be separate instruction caches and data caches. There may be multiple levels of caches.

[0003] If a processor requests an item from a cache and the item is present in the cache, the event is called a cache hit. If a processor requests an item from a cache and the item is not present in the cache, the event is called a cache miss. In the event of a cache miss, the requested item is retrieved from a lower level of the memory hierarchy. In many processor designs, the time required to access an item for a cache hit is one of the primary limiters for the clock rate of the processor. Therefore, optimization of access time for cache hits is critical for performance. There is an ongoing need for improvement in cache access time for computer processors.

[0004] Typically, a memory is organized into words (for example, 32 bits or 64 bits per word). Typically, the minimum amount of memory that can be transferred between a cache and a next lower level of the memory hierarchy is called a line, or sometimes a block. A line is typically multiple words (for example, 16 words per line). Memory may also be divided into pages (also called segments), with many lines per page. In some systems, page size may be variable.

[0005] In many modern computer memory architectures, a CPU produces virtual addresses that are translated by a combination of hardware and software to physical addresses, which access physical main memory. A group of virtual addresses may be dynamically assigned to each page. Virtual memory (paging or segmentation) requires a data structure, sometimes called a page table, that translates the virtual address to the physical address. To reduce address translation time, computers commonly use a specialized associative cache dedicated to address translation, commonly called a Translation Look-aside Buffer (TLB).

[0006] If a cache stores an entire line address along with the data and any line can be placed anywhere in the cache, the cache is said to be fully associative. How-ever, for a large cache in which any line can be placed anywhere, the hardware required to rapidly determine if an entry is in the cache (and where) may be very large and expensive. For large caches, a faster, space saving alternative is to use a subset of an address (called an index) to designate a line position within the cache, and then store the remaining set of more significant bits of each physical address (called a tag) along with the data. In a cache with indexing, an item with a particular address can be placed only within a set of lines designated by the index. If the cache is arranged so that the index for a given address maps to exactly one line in the subset, the cache is said to be direct mapped. If the index maps to more than one line in the subset, the cache is said to be set-associative. All or part of an address is hashed to provide a set index which partitions the address space into sets.

[0007] With direct mapping, when a line is requested, only one line in the cache has matching index bits. Therefore, the data can be retrieved immediately and driven onto a data bus before the system determines whether the rest of the address matches. The data may or may not be valid, but in the usual case where it is valid, the data bits are available on a bus before the system determines validity. With set-associative caches, it is not known which line corresponds to an address until the full address is compared. That is, in set-associative caches, the result of tag comparison is used to select which line of data bits within a set of lines is presented to the processor.

[0008] In a cache with a TLB, the critical timing path for a hit requires a sequence of four operations as follows: (1) a virtual tag must be presented to a Content Addressable Memory (CAM) in the TLB to determine the location of a corresponding physical tag in RAM in the TLB; (2) the physical tag must then be retrieved from the TLB RAM; (3) the physical tag from TLB RAM must then be compared to physical tags accessed from the tag section of the cache; (4) the appropriate data line must be selected. This sequence of four operations is preferably completed in a single phase of a clock cycle, and may in fact limit how short the phase can be. Each of the four operations takes time and a great deal of design effort is dedicated to making each operation as fast as possible. There is a need for improvement in the overall time for the critical time path for accesses to caches with TLB's.

SUMMARY OF THE INVENTION

[0009] A cache with a TLB is disclosed in which physical tags do not need to be retrieved from the TLB. Instead of storing physical tags in the cache, the cache stores the location within the TLB where the physical tag is stored. The TLB includes two CAM's. For a cache hit, one of the CAM's in the TLB generates a vector which specifies the location within the TLB where the physical address is stored. This vector is compared to

location vectors stored in the cache. This comparison of location vectors provides sufficient information to enable selection of one data line within a set without having to actually retrieve the physical address. As a result, a substantial time consuming operation (physical address retrieval) is removed from the critical time path of a cache hit. In addition, comparing location vectors rather than physical tags enables use of comparison logic which is faster and simpler than the digital comparators conventionally used.

[0010] If a system has variable page size, an additional structure is needed. Effectively, the additional structure is another CAM. The additional structure contains page offset bits (part of the address) and page mask bits (provided by the operating system) that designate which page offset bits are relevant. The index bits of the virtual address designate an entry in the additional structure.

[0011] A four-way set-associative cache is used for the example embodiment. However, the invention is applicable to any cache having a TLB.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012]

Figure 1 (prior art) is a block diagram schematic of an example set-associative cache.

Figure 2 is a block diagram schematic of an example set-associative cache in accordance with the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT OF THE INVENTION

[0013] Figure 1 illustrates an example of a prior art cache. Four-way set-associative caches are used for illustration, but the invention is applicable to any cache employing a TLB. A virtual address 100 comprises lower order index bits 102 and upper order tag bits 104. The index bits are typically the same for the virtual address and the physical address. The index bits are used to select one set of lines of data in a data section 106 of the cache. The output of data section 106 is four lines of data 108. The index bits are also used to select a set of physical tags in a tag section 110 of the cache. The output of the tag section 110 is four physical tags 112, each corresponding to one data line 108. The virtual tag bits 104 are used to select one entry in a CAM 116 within a TLB 114. The TLB 114 stores both virtual and physical tags. Note that the virtual tag 104 may not find a match in the CAM 116, in which case there is a TLB miss. Note also that multiple virtual tags may map to one physical tag. For a TLB hit, the selected CAM entry designates an address in TLB RAM 118 for a physical tag corresponding to the virtual tag 104. A physical tag is then retrieved from the TLB RAM 118.

Each of four digital comparators 120 then compares the physical tag from the TLB RAM 118 to a physical tag 112 from the tag section 110. A matching pair of physical tags indicates through logic 122 which one of four lines of data is selected by a multiplexer 124. Note that for the particular index bits there may not be a matching pair of physical tags, in which case there is a cache miss.

[0014] Figure 2 illustrates a 4-way set-associative cache illustrating an example embodiment of the invention. In figure 2, the index bits 102, the data section 106 of the cache, and the multiplexer 124 are the same as in figure 1. In the cache of figure 2, section 204 physically corresponds to section 110 of the cache of figure 1, but when a new line of data is placed into the cache of figure 2, instead of the physical tag being stored in section 204, a vector 212 (called a physical hit vector and described in more detail below) is stored in section 204.

[0015] In the cache of figure 2, TLB 210 has two CAM's, a physical CAM 206 containing physical tags and a virtual CAM 208 containing virtual tags. When a new virtual tag 202 is stored in the virtual CAM 208, the corresponding physical tag 203 is also available via the operating system, and the corresponding physical tag 203 is stored the physical CAM 206. The physical hit vector 212 has a binary "one" corresponding to each location in the physical CAM 206 that has the physical tag 203. For example, the physical hit vector

00000000000001000100000000000000

indicates that the physical tag 203, for this cache line, corresponding to the virtual tag 202, resides in the 14th entry and the 18th entry (for illustration arbitrarily numbered from the left starting with number one) of a 32-entry physical CAM 206. For example, the new virtual tag 202 and corresponding physical tag 203 may have been placed in the 18th entry of CAMs 208 and 206 respectively, and there may have already been a different virtual tag and the same corresponding physical tag in the 14th entry of CAMs 208 and 206 respectively. Upon entry of a new line into the cache, a binary physical hit vector 212, indicating the location of all the instances in the physical CAM 206 of the physical tag 203 of the new line, is generated by the physical CAM 206 and stored into cache section 204, at a row location determined by the index bits and at a column location determined by a set placement algorithm.

[0016] For a cache access for the cache of figure 2, virtual tag 202 is used by the virtual CAM 208 to generate a vector 214, called a virtual hit vector. If there is a TLB miss, the virtual hit vector 214 is all binary "zeros." If there is a TLB hit, the virtual hit vector 214 has a single binary "one" indicating the one location of the virtual tag 202 in the virtual CAM 208. (each virtual tag in the TLB must be unique, just as in the prior art TLB's).

[0017] For a cache access, the index bits 102 select a set of 4 physical hit vectors in cache section 204. Each of the 4 physical hit vectors in cache section 204 is compared (four comparators 216) to the one virtual hit vec-

tor 214 from the virtual CAM 208. In the cache of figure 1, for any given set of index bits 102, only one of the four selected physical tags in section 110 matches the physical tag from the TLB. Likewise, for the cache of figure 2, for a fixed page size (an example modification for variable page size is provided below), for any given set of index bits 102, only one of the four selected physical hit vectors in section 204 will have a binary "one" (all locations of the physical tag in the physical CAM 206) corresponding to the binary "one" in the virtual hit vector 214 (the one location of the virtual tag in the virtual CAM 208). For a fixed page size, a single pair of matching "ones" in the hit vectors then determines which data line is selected by multiplexer 124. Note that for a given set of index bits 102 there may not be any matching ones in the compared vectors, in which case there is a cache miss.

[0018]   The virtual hit vector 214 comprises information that must typically be generated internally by a TLB CAM in order to retrieve the physical tag from TLB RAM (for example, CAM 116 in figure 1). However, in the cache of figure 2, this information is used external to the TLB 210 and the physical tag from TLB 210 is not retrieved for a cache access. Eliminating the operation of retrieving the physical tag from TLB 210 eliminates an operation that takes a substantial amount of time in the critical time path for the cache.

[0019]   Digital comparators 120 in figure 1 comprise an EXCLUSIVE OR gate for each bit position followed by a large fan-in NOR gate (or logical equivalent). That is, in order to compare two physical tags, each bit position must be examined for two matching logical "ones" or two matching logical "zeros". In contrast, for physical hit vectors in figure 2, only a pair of matching logical "ones" is needed to determine a match. Therefore, comparators 216 may be simple AND gates followed by a large fan-in OR gate. AND gates are faster and smaller (simpler) than EXCLUSIVE OR gates and OR gates are much faster than NOR gates.

[0020]   One potential complication is that for some processor architectures some of the address bits may indicate page size, which may he variable. Typically, this is context or job dependent. For example, one page size may be used for a data base application and another page size for a graphics intensive application. As mentioned above, index bits are typically the same for the virtual address and the physical address. If page size changes, the number of physical index bits changes. Stated alternatively, there may be a set of bits in the address that sometimes are virtual and sometimes are physical. In the present patent document, we will call these bits page offset bits. For each page offset bit, the operating system provides a mask bit that indicates whether the corresponding page offset bit is physical or virtual. Then, the offset bits are included with the virtual tag sent to the TLB, and the mask bits indicate for each offset bit whether the offset bit should be excluded in the compare by the CAM in the TLB. For variable page size,

when a physical tag is stored in the physical CAM 206, the length of the stored tag effectively varies depending on how many of the page offset bit are included as a result of masking. However, the physical hit vector 212 represents the results of a comparison to only the physical tag 203 (excluding the page offset bits).

[0021]   For systems with variable page size, the cache in figure 2 includes a separate CAM structure 220. Each entry in structure 220 comprises the page offset bits for the address of a line and the corresponding mask bits. When a line is accessed in the cache, the virtual index bits select a row in the CAM structure 220. The CAM structure 220 compares the stored page offset bits (from offset bits 224) to the received page offset bits 222 (in the virtual address 200). For the CAM structure 220, the stored mask bits indicate for each offset bit location whether the offset bits should he included in the comparison. Likewise, physical CAM 206 includes the offset bits and mask bits with the physical tag, and for TLB 210, the stored mask bits indicate for each offset bit location whether the offset bits should be excluded from the comparison. Note that for each mask bit, a corresponding page offset bit will be included in the CAM 220 compare and excluded in the physical CAM 206 compare or vice versa. Since the physical hit vector 212 represents the results of a comparison to only the physical tag 203 (excluding the page offset bits) then for a large page size with variable page size, there may be multiple successful matches by comparators 216. CAM 220 compares the offset bits that were ignored in the generation of the physical hit vector 212. The outputs of CAM 220 are sets of lines of valid (included as imposed by the corresponding mask bits) offset bits. Comparators 226 then compare the outputs of CAM 220 to the page offset bits 222 of address 200. The outputs of the comparators 226 are then used to further qualify outputs of comparators 216. Then, logic 218 has two outputs indicating which one of four lines of data from section 106 is selected by multiplexer 124.

[0022]   The foregoing description of the present invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and other modifications and variations may be possible in light of the above teachings. The embodiment was chosen and described in order to best explain the principles of the invention and its practical application to thereby enable others skilled in the art to best utilize the invention in various embodiments and various modifications as are suited to the particular use contemplated. It is intended that the appended claims he construed to include other alternative embodiments of the invention except insofar as limited by the prior art.

Claims

1.   A method of accessing data in a cache, the method comprising the following steps:

receiving, by the cache, a virtual address (200), the virtual address including a virtual tag (202) and a virtual index (102), the virtual tag having a corresponding physical tag (203) separate from the virtual address;

generating a first vector (214), by a translation lookaside buffer (210), indicating a location in the translation lookaside buffer that contains a match to the virtual tag;

retrieving, by a first memory structure (204), from at least one location determined by the virtual index, at least one second vector, the second vector indicating whether at least one location in the translation lookaside buffer contains a tag that matches the physical tag;

comparing, by a digital comparator (216), the first vector and the second vector; and

using the result of comparing to select data from a second memory structure (106).

2. The method of claim 1, the virtual address further comprising page offset bits (222), the method further comprising:

comparing, by a third memory structure (220), the page offset bits (222) of the virtual address to page offset bits (224) stored in the third memory structure at a location determined by the virtual index; and

using the result of comparing by the third memory structure to qualify the result of comparing by the digital comparator.

3. A cache, the cache receiving a virtual address (200) comprising index bits (102) and tag bits (202), the cache comprising:

a translation-lookaside buffer (210), the translation lookaside buffer receiving the tag bits and generating a first vector (214) indicating a location in the translation lookaside buffer that contains a particular virtual address tag corresponding to the tag bits;

a memory structure (204), receiving the index bits, the memory structure containing second vectors, each second vector indicating at least one location of a physical address tag in the translation lookaside buffer, at least one second vector designated by the index bits;

at least one vector comparator (216) comparing the second vector from the memory structure to the first vector from the translation lookaside buffer.

4. The cache of claim 3, the virtual address further comprising page offset bits (222), the cache further comprising:

a second memory structure (220), receiving the index bits, the second memory structure containing stored page offset bits (224);

at least one offset comparator (226) receiving stored page offset bits from the second memory structure as designated by the index bits and receiving page offset bits (222) from the virtual address;

at least one logic gate receiving an output of the offset comparator and an output of the vector comparator.
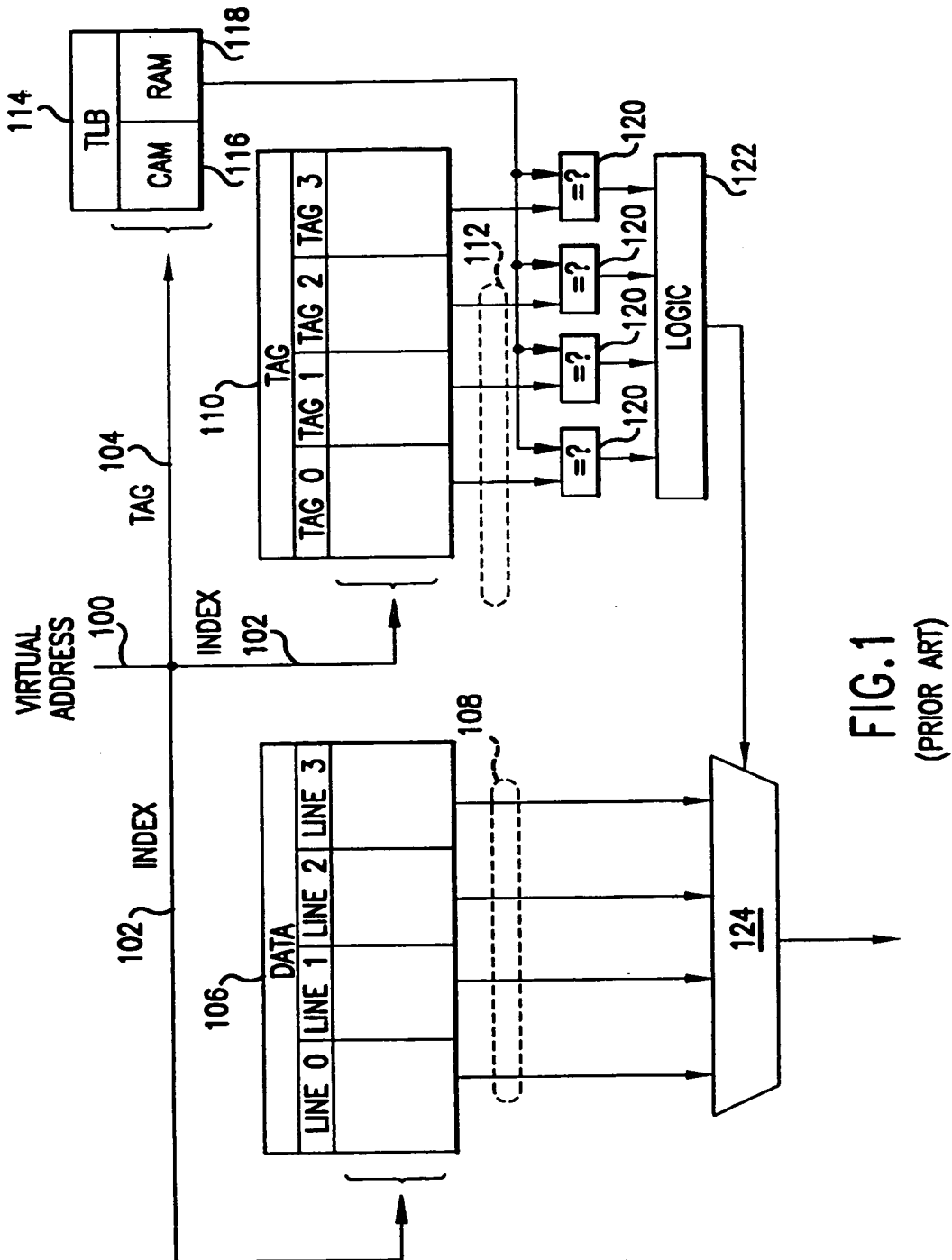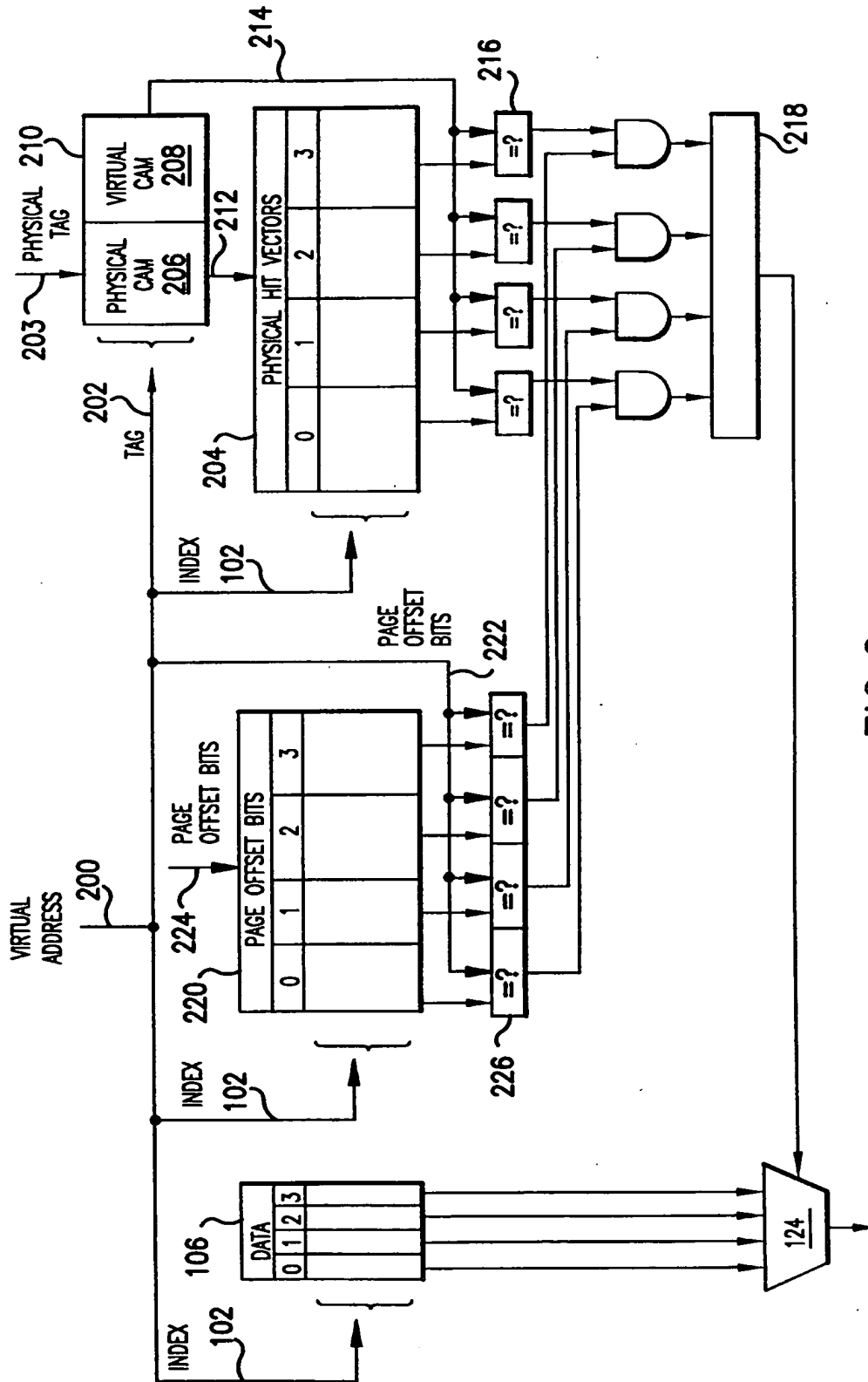
FIG.1
(PRIOR ART)

FIG.2

European Patent
Office

EUROPEAN SEARCH REPORT

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	US 4 969 122 A (JENSEN ERIC H) 6 November 1990 * the whole document *	1,3	G06F12/10
X	US 5 133 058 A (JENSEN ERIC H) 21 July 1992	1,3	
A	* column 6, line 42 – column 8, line 59; figures 4,5 *	2,4	
			TECHNICAL FIELDS SEARCHED (Int.Cl.6)
			G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	4 February 1999	Nielsen, O

EPO FORM 1503 03.82 (P04C01)

EP 0 911 737 A1

ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.   EP 98 10 8916

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

04-02-1999

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4969122 A	06-11-1990	CA 2022529 A,C FR 2651050 A	22-02-1991 22-02-1991
US 5133058 A	21-07-1992	AU 623517 B AU 5377690 A CA 2022656 A	14-05-1992 21-03-1991 19-03-1991